Mining Web Navigation Patterns with Dynamic Thresholds for Navigation Prediction

Jia-Ching Ying¹, Chu-Yu Chin¹,² and Vincent S. Tseng¹

¹Institute of Computer Science and Information Engineering
National Cheng-Kung University,
Tainan, Taiwan, R.O.C.

²Telecommunication Laboratories ChungHwa Telecom Co., Ltd.,
Taiwan, R.O.C.

jashying@gmail.com, cchuyu@cht.com.tw and tsengsm@mail.ncku.edu.tw

Abstract—Discovering web navigation patterns is an important issue in web usage mining with various applications like navigation prediction and improvement of website management. Since web site structure is always changed, we need not only consider the frequency of click behavior but also web site structure to mine web navigation patterns for navigation prediction. To reduce the overhead of dynamically mining the web navigation patterns from the web data, a dynamic mining approach is needed by using the previous mining results and computing new patterns just from the inserted or deleted part of the web data. In this paper, we propose a special data structure named Ideal-Tree (Inverted-database Expectable Tree) to avoid the effort of scanning database. Meanwhile, an efficient mining algorithm named Ideal-Tree-Miner is proposed for mining web navigation patterns with dynamic thresholds. Based on the discovered patterns, we also give a navigation prediction model. The experimental results show that our prediction model outperforms other approaches substantially in terms of Precision, Recall, and F-measure.

Keywords—Web mining; Incremental mining; Web navigation pattern; Navigation prediction

I. INTRODUCTION

In recent years, vast amount of data would be easily produced and collected from the web environment because of the growth of Internet. Hence, how to discover the useful information and knowledge efficiently from these vast amounts of web data has become a more and more important topic recently. Web navigation pattern mining [1][2] is present to improve the web services by extracting useful information and knowledge from vast amount of web data.

However, the web data grows rapidly and some of the user sequences may be out of date over time. Therefore, the web navigation patterns should be updated when the new web data is inserted into original web navigation sequence database. Therefore, the web navigation patterns should be updated when the new web data are inserted into original web navigation sequence database [5][7][8][9][15]. In addition, the frequent web navigation patterns may not always be the interesting web navigation patterns because the access design of web pages is not the same [9]. Therefore, a flexible model of mining navigation patterns with dynamic thresholds is needed. Unfortunately, to the best of our knowledge, there is no any web navigation pattern mining algorithm which considers dynamic thresholds. Besides, there is no any existing work could make navigation prediction in the incremental mining environment.

In this paper, we propose a novel efficient incremental algorithm for mining web navigation patterns with dynamic thresholds named id-Matrix-Miner (Inverted-database Matrix Miner), for re-mining all the web navigation patterns when the database is updated. Furthermore, a novel structure we proposed named id-Matrix (Inverted-database Matrix) and inverted file database are selected as our storage structure for id-Matrix-Miner to store original web dataset. The id-Matrix is kept in memory and the projected database is store in hard disk. That is, all of the necessary information in original web dataset is stored in these two kinds of storage. Hence, id-Matrix-Miner can avoid unnecessary scanning of the original web dataset. Thus, I/O cost could be frugal. Through a series of experiments, id-Matrix-Miner is about 3–5 times faster than IDTM algorithm in most case. Moreover, based on the id-Matrix-Miner, we propose a navigation prediction model which could handle such incremental mining environment.

The rest of this paper is organized as follows. In Section 2, we give the related work. Our mining method, id-Matrix-Miner, is given in Section 3. In section 4 and 5, we describe the Pattern Tree Construction and Navigation prediction. The experimental evaluation results are described in details in Section 6. In Section 7, a brief conclusion with future work is given.

II. RELATED WORK

Navigation Pattern Mining. In fact, web navigation pattern has a great resemblance to web traversal pattern. The most significant difference between the tow kinds of patterns is that web navigation patterns consist of contiguous web page references. Thus, the traversal pattern mining algorithms are easy to apply to the navigation pattern mining. Most of studies had discussed incremental web traversal or navigation patterns mining problems which consider the
uniform threshold [5][7][8][15] and dynamic threshold [9]. In [5], El-Sayed et al. presented FS-Miner algorithm that preserves some necessary information in a Matrix structure, named frequent sequence tree structure (FS-tree). In [7] and [8], Lee proposed an IneWTP algorithm uses a lattice structure, named extended lattice structure, to keep the candidate sequences whose support counts are greater than zero. Besides, the sequence id also is stored in the lattice structure to improve the efficiency during incremental step. In [15], Ying proposed an incremental mining algorithm for traversal patterns. The paper used a matrix named projected-database link matrix to store necessary information.

**Dynamic Mining.** In IDTM [9], a Markov chains and the underlying website structure are used to calculate the expected value recurrence time of web page \( p \), denoted by \( u_p \), and identify dynamic mining model as the following formula:

\[
\begin{align*}
\min_{\sup}(p) &= m \times |D|/u_p, & \text{if } m \times |D|/u_p \geq S_{th} \\
\min_{\sup}(p) &= S_{th}, & \text{if } m \times |D|/u_p \leq S_{th}
\end{align*}
\]

where \( |D| \) is the size of the database and \( m \in [0, 1] \) is a parameter to determine the relationship between the interestingness supports of Web pages and their expected value recurrence time. In addition, the threshold of a sequence \( <p_1p_2...p_n> \) is \( \min_{\sup}(p_{i=1,2,3,...n}) \). Although IDTM algorithm [9] can deal with the dynamic threshold mining problem, it is very time-consuming for incremental step, because it is an apriori-like algorithm.

When a new web navigation pattern with length \( k \) is discovered from the updated database, several new candidate sequences with length \( k+1 \) may be generated. Hence the algorithms must scan whole web data (new and original web dataset) once.

**Navigation Prediction.** Many data mining studies have discussed the Navigation Prediction problems for predicting the next location where a mobile user moves to. Personal-based prediction [6] [13] [14] and general-based prediction [10] [16] [17] are two approaches often adopted in this problem domain. The personal-based prediction approach considers movement behavior of each individual as independent and thus uses only the movements of an individual user to predict his/her next location. On the contrary, the general-based prediction makes a prediction based on the common movement behavior of general mobile users. In [6], Jeung et al. propose an innovative approach which forecasts future locations of a user by combining predefined motion functions, i.e., linear or non-linear models that capture object movements as sophisticated mathematical formulas, with the movement patterns of the user, extracted by a modified version of the Apriori algorithm. In [13], Yavas et al. mine the movement patterns of an individual user to form association rules and use these rules to make location prediction. Additionally, they consider the support and confidence in selecting the association rules for making predictions. In [14], Ye et al. propose a novel pattern, called Individual Life Pattern, which is mined form individual trajectory data, and they uses such pattern to describe and model the mobile users’ periodic behaviors. In [10], Monreale et al. proposes a method aiming to predict with a certain level of accuracy the next location of a moving object. The movement patterns extracted for prediction covers three different movement behaviors, including order of locations, travel time, and frequency of user visits. In [16], Zheng et al. uses a HITS-based model to mine users’ interesting location and detect users’ travel sequence to make locations prediction, and in [17], they consider the location correlation for generating the users’ interesting locations and travel sequence. Note that the above-mentioned prediction methods are based on geographic information only for predicting the next location where a mobile user moves to. On the contrary, our proposal predicts the next clicked page of a user based on navigation pattern which discovered from click sequence.

**III. NAVIGATION PATTERN MINING**

In this section, we describe the proposed method of web navigation patterns mining for dynamic thresholds. First, the id-Matrix (Inverted-database Matrix) and Inverted-database are proposed in subsection 3.1. Next, an incremental mining algorithm named id-Matrix-Miner is proposed in subsection 3.2.

**A. id-Matrix and Inverted-database**

The objective of the id-Matrix (Inverted-database Matrix) and Inverted-databases is to store the necessary information to avoid scanning original database such that the efficiency of id-Matrix-Miner could be improved. The Inverted-database Matrix is kept in memory, and the Inverted-database is store in hard disk.

Each entry of the Inverted-database Matrix consists of the following three components: 1) a link which points to an inverted database, 2) count of a navigation pattern with length 2. Now we take an Example to describe the Inverted-database Matrix. Table 1 shows an original database example which contains 5 sequences, where the column of id represents the id of a maximal forward references [3] and the column of sequence represents the maximal forward references. After one pass of the database scan, we have the Inverted-database Matrix and several inverted-databases as shown in Figure 1. At the same time, the entry whose count is increased will be marked.

**TABLE I. ORIGINAL DATABASE.**

<table>
<thead>
<tr>
<th>id</th>
<th>sequence</th>
<th>id</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;ABCD&gt;</td>
<td>4</td>
<td>&lt;CAD&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;CABD&gt;</td>
<td>5</td>
<td>&lt;AD&gt;</td>
</tr>
<tr>
<td>3</td>
<td>&lt;AB&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
B. id-Matrix-Miner

After obtaining the id-Matrix (Inverted-database Matrix) and Inverted-database, next step is to mine the web navigation patterns from the id-Matrix and the Inverted-database. Consider web navigation sequence database as Table 1 and the threshold of each web page as Table 2, for each entry (i,j) which is marked in the id-Matrix, id-Matrix-Miner will perform four main steps on the entry:

1) Check pattern \(<i,j>\). We first check whether the count of entry \((i,j)\) is equal to or greater than \(\min\{\min\_sup(i), \min\_sup(j)\}\). The pattern \(<i,j>\) will be outputted if the count of entry \((i,j)\) is equal to or greater than \(\min\{\min\_sup(i), \min\_sup(j)\}\). For example, in Figure 1, the entry \((A,B)\) is greater than \(\min\{\min\_sup(A), \min\_sup(B)\}\). Thus, the pattern \(<AD>\) should be outputted.

2) Load the Inverted-database in main memory. We then load the \(<i,j>-\text{Inverted-database}\) in main memory if the count of entry \((i,j)\) is equal to or greater than \(\min\{\min\_sup(p)\}\) for all web page \(p\) in the \(<i,j>-\text{Inverted-database}\). For example, in Figure 1, the entry \((A,B)\) is greater than \(\min\{\min\_sup(C), \min\_sup(D)\}\). Thus, the \(<AB>-\text{Inverted-database}\) should be loaded into main memory.

3) Count the single web page. After the Inverted-database is loaded in main memory, for each web page in the Inverted-database, we then count the number of location is the first by one pass of the Inverted-database scan. If the count of web page \(X\) is equal to or greater than \(\min\{\min\_sup(X)\cup\{\min\_sup(p)\}\}_{1\leq l\leq n}\) and \(<p_{p_2...p_l}>\) is the prefix of the Inverted-database, the pattern \(<p_{p_2...p_l}X>\) will be outputted. For example, in Figure 2, the web page \(D\) has only one location is first, \((2,1)\). So the count of web page \(D\) is 1 and equal to \(\min\{\min\_sup(C), \min\_sup(A), \min\_sup(B)\}\). Thus, the pattern \(<ABD>\) should be outputted.

4) Produce local Inverted-database. We then perform a second scan of the Inverted-database to obtain the local Inverted-databases. For each nonempty local Inverted-database, if the count of its prefix is equal to or greater than \(\min\{\min\_sup(p)\}\) for all web page \(p\) in the local Inverted-database \({}\), we make the local Inverted-database as input and go to the step 3 (count the single web page). For example, in Figure 3, the count of \(<ABC>\) is greater than \(\min\{\min\_sup(D)\}\). Thus, we will make the \(<ABC>-\text{Inverted-database}\) as input and go to the step 3.

We then address how to perform id-Matrix-Miner on them. After one pass of the database scan, we have the Inverted-database Matrix and several Inverted-databases as shown in Figure 4. At the same time, all the entry whose count is increased will be mark. Then, we only need to perform id-Matrix-Miner on the entry which is marked. For example, in Figure 4, only entries (A,B), (A,C), (A,D), (B,A), (B,D), and (C,A) should be visited.
IV. PATTERN TREE CONSTRUCTION

After web navigation pattern mining, such patterns could provide several decision rules for location prediction. For example, if a pattern \(<A, B, C, D>\) is discovered from a web data, we can predict that he/she may browse page C after browsing page A and then B. Therefore, by matching a mobile user’s recent click behavior to his/her web navigation patterns, we can predict his/her next browsed page. However, it is clear to observe that the longer pattern we mine the more subsequences will be generated due to the downward closure property. It leads to a loss of efficiency because all the subsequences of a long pattern need to be considered in the next location prediction. For example, the subsequences of the pattern \(<A, B, C>\) are \(<A>\), \(<B>\), \(<C>\), \(<A, B>\), \(<B, C>\), and \(<A, C>\). It is very time-consuming to match the current move of a mobile user to all his/her web navigation patterns one by one. To make the prediction phase efficient, we adopted a prefix tree, named web navigation pattern tree (WNP-Tree), to compactly represent a collection of web navigation patterns. Note that the path of a WNP-Tree indicates a decision rule. The WNP-Tree is a kind of decision tree, where each node \(v\) consists of tree element, web page set, support, and children.

The WNP-TreeBuilding algorithm, shown in Figure 5, describes how to build the WNP-Tree from a web navigation patterns set (WNP-Set). In the following, we introduce the notion of prefix of a web navigation pattern. For simplicity, we consider a web navigation pattern as a sequence of web page labels. Each web navigation pattern belonging to the WNP-Set is inserted into the WNP-Tree. Intuitively, given a web navigation pattern WNP, we search the tree for the path corresponding to the longest prefix of WNP. Next, we append a branch to cover the remaining elements of WNP in this path. A web navigation pattern is appended to a path in the tree if this path is a prefix of web navigation pattern. When the pattern is appended to a path, the support value will be updated if the support value of pattern is greater than the support value of the node (see Line 5 to 9 of Figure 5). The CreateNode(web page, support, children) function returns the node which stores the web page label, support value, and children list. The appendChild(child) procedure appends another node to the children list of a node (see Line 10 to 13 of Figure 5).

As shown in Table 2, the web navigation pattern is mined from the web navigation dataset. Figure 6 shows the corresponding web navigation pattern tree. The path with only one node will be eliminated from the pattern tree, e.g., the pattern \(<C>\) is not shown in the pattern tree. Since the tree can be modified in real time, this tree based storage could handle the incremental mining environment well. When the new patterns are obtained or old patterns are deleted, we can easily modify the tree for navigation prediction.

V. NAVIGATION PREDICTION

Given a web user, the prediction model predicts her next browsed page on her own web navigation pattern tree. Given this pattern tree, the browsing information (i.e., the web navigation patterns) user belong can be utilized in the prediction. Thus, given the browsing sequence of a user’s recent click behavior, we compute the best matching scores of candidate paths in these two pattern trees. The matching scores are computed by formula as follows:

\[
Score(P, S) = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha^{P_{ij}} \times mScore(P_i, S_j),
\]

where \(mScore(P_i, S_j) = \begin{cases} P_i . support , & \text{if } S_j \text{ is matching to } P_i \\ 0, & \text{otherwise} \end{cases}\)

User current browsing page sequence: \(<C, A, B>\)

\(k=1\) \(k=2\) \(k=3\)

Figure 5. WNP-TreeBuilding algorithm.

Figure 6. An example of WNP-Tree.

Figure 7. An example of path matching.
In order to simplify the matching process, the current user’s recent click behaviors are transformed into a web page sequence. Moreover, since the web page sequence may consist of too many web pages, it is very time consuming to consider all possible subsequences of the web page sequence in the matching step. Therefore, we propose a partial matching strategy which does not consider all the possible subsequences of the web page sequence. Instead, the score of click behavior captures three heuristics: 1) outdated browsed pages may potentially deteriorate the precision of predictions; 2) more recent browsed pages potentially have more important impacts on predictions; and 3) the matching path with a higher support and a higher length may provide a greater confidence for predictions. Given a mobile user’s web page sequence \( S \) and a matching path \( P \) in WNP-Tree, we propose a weighted scoring function, \( mScore(P, S) \), as defined in Equation (2).

<table>
<thead>
<tr>
<th>matching paths</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>C ( \rightarrow ) A ( \rightarrow ) B</td>
<td>0</td>
</tr>
<tr>
<td>A ( \rightarrow ) B</td>
<td>( 0.8 \times 0.667 + 0.667 = 1.2 )</td>
</tr>
<tr>
<td>B</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Finally, we use Equation (2) to evaluate the score of each page. We predict the children of the candidate path with the highest score as the answer. Take Figure 7 as an example, the match path and its score is shown as table 3. The path A \( \rightarrow \) B has greatest score. Thus we predict the user may brows the web page C. Note that if the path with the highest score has no children, we predict the children of the candidate path with the second highest score, and so on.

VI. COPYRIGHT FORMS AND REPRINT ORDERS

In this section, we conducted a series of experiments to evaluate the performance of the proposed \( id\text{-Matrix-Miner} \) under various conditions. All of the experiments implemented in Java JDK 1.6 on an Intel Core Quad CPU Q6600 2.40GHz machine with 1GB of memory running Microsoft Windows XP.

A. 6.1 KDD Cup 2000 Dataset

The KDD Cup 2000 dataset can be downloaded from the URL: http://cobweb.ecn.purdue.edu/ KDDCUP/. The dataset contains 777,480 clicks, 234,954 sessions and 137 different Request Templates. We treat Request Template as web page and group the dataset by session id. In each group, the web pages are order by referred time as session sequence. Thus we have 234,954 session sequences and 137 different web pages. The average length of the sequences is 3.287. We divide the dataset into two parts: one with size 188K as original user session database and another with size 47K as the inserted dataset.

B. 6.2 Mining Efficiency

In this section, we conducted a series of experiments to evaluate the performance for the proposed \( id\text{-Matrix-Miner} \) under various conditions. The parameter \( m \)

1) 6.2.1 Impact of varied parameter \( m \)

This experiment analyzes the execution time when parameter \( m \) varies. Figure 8 shows that \( id\text{-Matrix-Miner} \) outperforms \( IDTM \) algorithm in terms of execution time. We observed that the parameter \( m \) is set smaller, the better result of \( id\text{-Matrix-Miner} \). The reason is that the most patterns are mined based on the model of a uniform support threshold when the \( m \) is set very small. Therefore, the prune strategy in \( IDTM \) algorithm can not bring significant improvement.

2) 6.2.2 Impact of varied the inserted database size

This experiment analyzes the execution time when the inserted database size varies. Figure 6 shows that \( id\text{-Matrix-Miner} \) outperforms \( IDTM \) algorithm in terms of execution time. We observed that the execution time of both \( id\text{-Matrix-Miner} \) and \( IDTM \) algorithm is linear growth with inserted database size. Because that when the parameter is great enough, both of two nodes also algorithms can prune many times of database scan. Therefore, the algorithms will converge linear time algorithms. Overall, \( id\text{-Matrix-Miner} \) algorithm is about 3–5 times faster than \( IDTM \) algorithm in most case.

C. 6.2 Prediction Effectiveness

In this section, we conducted a series of experiments to evaluate the effectiveness for the proposed prediction model in terms of Precision, Recall, and F-measure. The followings are the main measurements for the experimental evaluation. The Precision, Recall, and F-measure are defined as Equations (3), (4), and (5), where \( p^+ \) and \( p^- \) indicate the number of correct predictions and incorrect predictions, respectively, and \( |R| \) indicates the total number of web sequence.

\[
\text{Precision} = \frac{p^+}{p^+ + p^-} \quad (3)
\]

\[
\text{Recall} = \frac{p^+}{|R|} \quad (4)
\]

\[
\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5)
\]
We can observe that dynamic navigation pattern does not outperform traditional navigation pattern in terms of precision but in terms of recall and F-measure. The reason is that dynamic navigation pattern mining will obtain some patterns which can not be discovered by traditional navigation pattern mining. Such pattern may be useful for prediction but not at all. So the precision may slightly decrease and the recall will significantly increase.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed an efficient incremental data mining algorithm named id-Matrix-Miner for mining web navigation patterns with dynamic threshold. We also proposed a new data structure, id-Matrix, for storing the useful information so as to avoid re-mining the original database. Hence, the web navigation patterns can be discovered efficiently when the navigation sequence database is updated. The Experimental results show that id-Matrix-Miner algorithm is about 3-5 times faster than IDTM algorithm in most case. Moreover, our prediction model outperforms traditional navigation pattern based model in terms of Recall and F-measure.

ACKNOWLEDGMENT

This research was partially supported by Telecommunication Laboratories, Chunghwa Telecom, Taiwan and by National Science Council, Taiwan, under grant no. NSC100-2631-H-006-001.

REFERENCES